

Лабораторная работа №7

Построение нейронных сетей в Matlab

Нейронные сети (NN - Neural Networks) широко используются для решения разнообразных задач. Среди развивающихся областей применения NN - обработка аналоговых и цифровых сигналов, синтез и идентификация электронных цепей и систем. Основы теории и технологии применения NN широко представлены в пакете MATLAB. В этой связи особо следует отметить последнюю версию пакета - MATLAB 6.0, где впервые представлен GUI (Graphical User Interface - графический интерфейс пользователя) для NN - NNTool.

Примерами применения технологии нейронных сетей для цифровой обработки сигналов являются: фильтрация, оценка параметров, детектирование, идентификация систем, распознавание образов, реконструкция сигналов, анализ временных рядов и сжатие. Упомянутые виды обработки применимы к разнообразным видам сигналов: звуковым, видео, речевым, изображения, передачи сообщений, геофизическим, локационным, медицинских измерений (кардиограммы, энцефаллограммы, пульс) и другим.

В данной лабораторной работе дано описание NNTool и показана техника его применения в ряде задач синтеза цепей и цифровой обработки сигналов.

Обработка сигналов в технологиях NN выполняется с помощью либо NN без памяти, либо NN с памятью. И в том и другом случаях ключевым элементом является NN без памяти. Подобная роль определяется тем обстоятельством, что при использовании нейронов с определёнными функциями активации (передаточными характеристиками) NN является универсальным аппроксиматором. Последнее означает, что в заданном диапазоне изменения входных переменных NN может с заданной точностью воспроизводить (моделировать) произвольную непрерывную функцию этих переменных.

После того как структура NN выбрана, должны быть установлены её параметры. Выбор структуры NN и типов нейронов - самостоятельный и весьма непростой вопрос, который здесь мы обсуждать не будем. Что же касается значений параметров, то, как правило, они определяются в процессе решения некоторой оптимизационной задачи. Эта процедура в теории NN называется обучением.

Графический интерфейс пользователя NNTool позволяет выбирать структуры NN из обширного перечня и предоставляет множество алгоритмов обучения для каждого типа сети.

В работе рассмотрены следующие вопросы, относящиеся к работе с NNTool:

- назначение графических управляющих элементов;
- подготовка данных;
- создание нейронной сети;
- обучение сети;

- прогон сети.

Управляющие элементы NNTool

Чтобы запустить NNTool, необходимо выполнить одноимённую команду в командном окне MATLAB:

```
>> nntool
```

после этого появится главное окно NNTool, именуемое "Окном управления сетями и данными" (Network/Data Manager) (рис. 1).

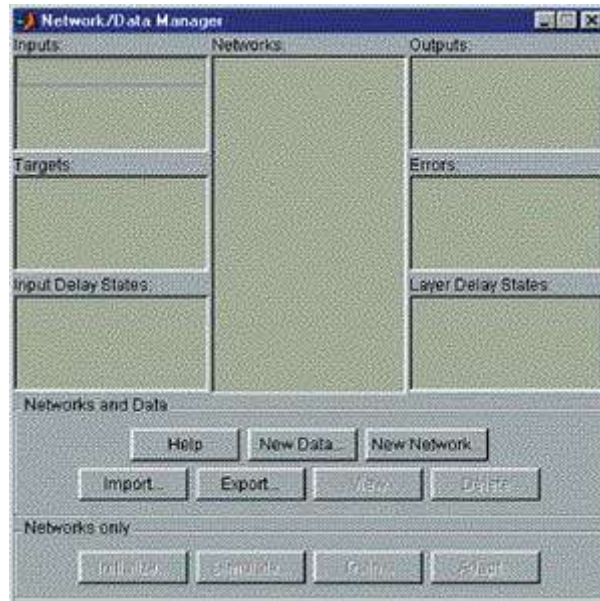


Рисунок 1. Главное окно NNTool

Панель "Сети и данные" (Networks and Data) имеет функциональные клавиши со следующими назначениями:

- Помощь (Help)- краткое описание управляющих элементов данного окна;
- Новые данные (New Data...)- вызов окна, позволяющего создавать новые наборы данных;
- Новая сеть (New Network...)- вызов окна создания новой сети;
- Импорт (Import...)- импорт данных из рабочего пространства MATLAB в пространство переменных NNTool;
- Экспорт (Export...)- экспорт данных из пространства переменных NNTool в рабочее пространство MATLAB;
- Вид (View)- графическое отображение архитектуры выбранной сети;
- Удалить (Delete)- удаление выбранного объекта.

На панели "Только сети" (Networks only) расположены клавиши для работы исключительно с сетями. При выборе указателем мыши объекта любого другого типа, эти кнопки становятся неактивными.

При работе с NNTool важно помнить, что клавиши View, Delete, Initialize, Simulate, Train и Adapt (изображены на рис. 1 как неактивные) действуют применительно к тому объекту, который отмечен в данный момент выделением. Если такого объекта нет, либо над выделенным объектом невозможно произвести указанное действие, соответствующая клавиша неактивна.

Рассмотрим создание нейронной сети с помощью NNTool на примере

Пример 1.

Пусть требуется создать нейронную сеть, выполняющую логическую функцию "И".

Создание сети

Выберем сеть, состоящую из одного персептрона с двумя входами. В процессе обучения сети на её входы подаются входные данные и производится сопоставление значения, полученного на выходе, с целевым (желаемым). На основании результата сравнения (отклонения полученного значения от желаемого) вычисляются величины изменения весов и смещения, уменьшающие это отклонение.

Итак, перед созданием сети необходимо заготовить набор обучающих и целевых данных. Составим таблицу истинности для логической функции "И", где P1 и P2 - входы, а A - желаемый выход (табл. 1).

Таблица 1. Таблица истинности логической функции "И"

P1	P2	A
0	0	0
0	1	0
1	0	0
1	1	1

Чтобы задать матрицу, состоящую из четырёх векторов-строк, как входную, воспользуемся кнопкой New Data. В появившемся окне следует произвести изменения, показанные на рис. 2, и нажать клавишу "Создать" (Create).

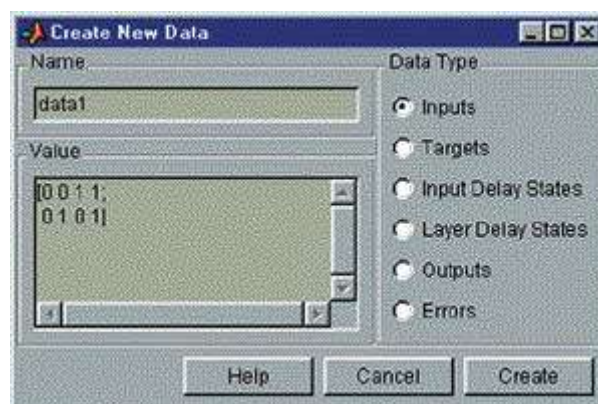


Рисунок 2. Задание входных векторов

После этого в окне управления появится вектор data1 в разделе Inputs. Вектор целей задаётся схожим образом (рис. 3).

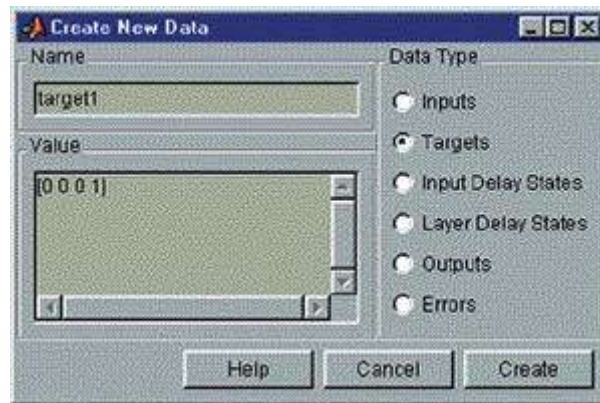


Рисунок 3. Задание целевого вектора

После нажатия на Create в разделе Targets появится вектор target1.

Данные в поле "Значение" (Value) могут быть представлены любым понятным MATLAB выражением. К примеру, предыдущее определение вектора целей можно эквивалентно заменить строкой вида

`bitand([0 0 1 1], [0 1 0 1]).`

Теперь следует приступить к созданию нейронной сети. Выбираем кнопку New Network и заполняем форму, как показано на рис. 4.

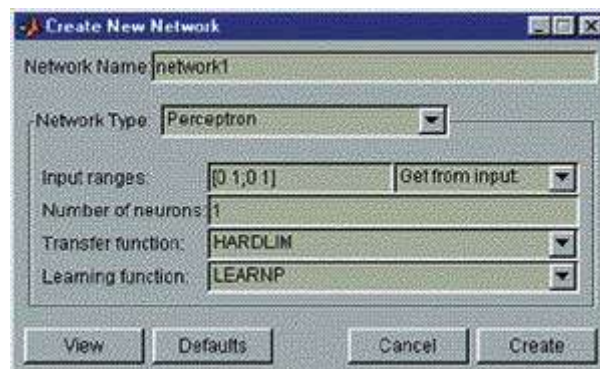


Рисунок 4. Окно "Создание сети"

При этом поля несут следующие смысловые нагрузки:

- Имя сети (Network Name) - это имя объекта создаваемой сети.
- Тип сети (Network Type)- определяет тип сети и в контексте выбранного типа представляет для ввода различные параметры в части окна, расположенной ниже этого пункта. Таким образом, для разных типов сетей окно изменяет своё содержание.
- Входные диапазоны (Input ranges)- матрица с числом строк, равным числу входов сети. Каждая строка представляет собой вектор с двумя элементами: первый - минимальное значение сигнала, которое будет подано на соответствующий вход сети при обучении, второй - максимальное. Для упрощения ввода этих значений предусмотрен выпадающий список "Получить из входа" (Get from input), позволяющий автоматически сформировать необходимые данные, указав имя входной переменной.
- Количество нейронов (Number of neurons)- число нейронов в слое.

- Передаточная функция (Transfer function)- в этом пункте выбирается передаточная функция (функция активации) нейронов.
- Функция обучения (Learning function)- функция, отвечающая за обновление весов и смещений сети в процессе обучения.

С помощью клавиши "Вид" (View) можно посмотреть архитектуру создаваемой сети (рис. 5). Так, мы имеем возможность удостовериться, все ли действия были произведены верно. На рис. 5 изображена персептронная сеть с выходным блоком, реализующим передаточную функцию с жёстким ограничением. Количество нейронов в слое равно одному, что символически отображается размерностью вектора-столбца на выходе слоя и указывается числом непосредственно под блоком передаточной функции. Рассматриваемая сеть имеет два входа, так как размерность входного вектора-столбца равна двум.

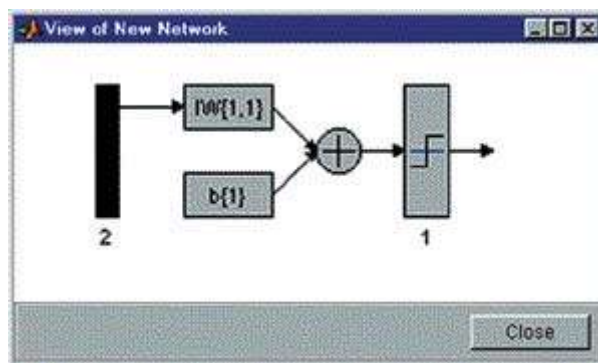


Рисунок 5. Предварительный просмотр создаваемой сети

Итак, структура сети соответствует нашему заданию. Теперь можно закрыть окно предварительного просмотра, нажав клавишу "Закрыть" (Close), и подтвердить намерение создать сеть, нажав "Создать" (Create) в окне создания сети.

В результате проделанных операций в разделе "Сети" (Networks) главного окна NNTool появится объект с именем network1.

Обучение

Наша цель - построить нейронную сеть, которая выполняет функцию логического "И". Очевидно, нельзя рассчитывать на то, что сразу после этапа создания сети последняя будет обеспечивать правильный результат (правильное соотношение "вход/выход"). Для достижения цели сеть необходимо должным образом обучить, то есть подобрать подходящие значения параметров. В MATLAB реализовано большинство известных алгоритмов обучения нейронных сетей, среди которых представлено два для персептронных сетей рассматриваемого вида. Создавая сеть, мы указали LEARNP в качестве функции, реализующей алгоритм обучения (рис. 4).

Вернёмся в главное окно NNTool. На данном этапе интерес представляет нижняя панель "Только сети" (Networks only). Нажатие любой из клавиш на этой панели вызовет окно, на множестве вкладок которого представлены параметры сети, необходимые для её обучения и прогона, а также отражающие текущее состояние сети.

Отметив указателем мыши объект сети network1, вызовем окно управления сетью нажатием кнопки Train. Перед нами возникнет вкладка "Train" окна свойств сети, содержащая, в свою очередь, ещё одну панель вкладок (рис. 6). Их главное назначение - управление процессом обучения. На вкладке "Информация обучения" (Training info) требуется указать набор обучающих данных в поле "Входы" (Inputs) и набор целевых данных в поле "Цели" (Targets). Поля "Выходы" (Outputs) и "Ошибки" (Errors) NNTool заполняет автоматически. При этом результаты обучения, к которым относятся выходы и ошибки, будут сохраняться в переменных с указанными именами.



Рисунок 6. Окно параметров сети, открытое на вкладке "обучение" (Train)

Завершить процесс обучения можно, руководствуясь разными критериями. Возможны ситуации, когда предпочтительно остановить обучение, полагая достаточным некоторый интервал времени. С другой стороны, объективным критерием является уровень ошибки.

На вкладке "Параметры обучения" (Training parameters) для нашей сети (рис. 7) можно установить следующие поля:

- Количество эпох (epochs)- определяет число эпох (интервал времени), по прошествии которых обучение будет прекращено.
- Эпохой называют однократное представление всех обучающих входных данных на входы сети.
- Достижение цели или попадание (goal)- здесь задаётся абсолютная величина функции ошибки, при которой цель будет считаться достигнутой.
- Период обновления (show)- период обновления графика кривой обучения, выраженный числом эпох.
- Время обучения (time)- по истечении указанного здесь временного интервала, выраженного в секундах, обучение прекращается.



Рисунок 7. Вкладка параметров обучения

Принимая во внимание тот факт, что для задач с линейно отделимыми множествами (а наша задача относится к этому классу) всегда существует точное решение, установим порог достижения цели, равный нулю. Значения остальных параметров оставим по умолчанию. Заметим только, что поле времени обучения содержит запись Inf, которая определяет бесконечный интервал времени (от английского Infinite - бесконечный).

Следующая вкладка "Необязательная информация" (Optional Info) показана на рис. 8.



Рисунок 8. Вкладка необязательной информации

Рассмотрим вкладку обучения (Train). Чтобы начать обучение, нужно нажать кнопку "Обучить сеть" (Train Network). После этого, если в текущий момент сеть не удовлетворяет ни одному из условий, указанных в разделе параметров обучения (Training Parameters), появится окно, иллюстрирующее динамику целевой функции - кривую обучения. В нашем случае график может выглядеть так, как показано на рис. 9. Кнопкой "Остановить обучение" (Stop Training) можно прекратить этот процесс. Из рисунка видно, что обучение было остановлено, когда функция цели достигла установленной величины (goal = 0).

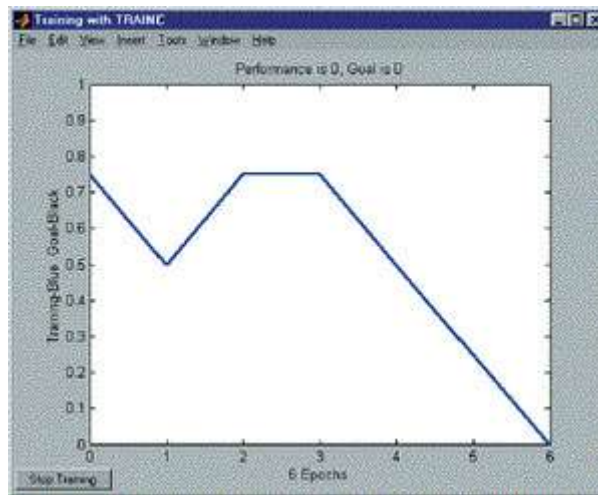


Рисунок 9. Кривая обучения

Следует отметить, что для персептронов, имеющих функцию активации с жёстким ограничением, ошибка рассчитывается как разница между целью и полученным выходом.

Итак, алгоритм обучения нашёл точное решение задачи. В методических целях убедимся в правильности решения задачи путём прогона обученной сети. Для этого необходимо открыть вкладку "Прогон" (Simulate) и выбрать в выпадающем списке "Входы" (Inputs) заготовленные данные. В данной задаче естественно использовать тот же набор данных, что и при обучении data1. При желании можно установить флажок "Задать цели" (Supply Targets). Тогда в результате прогона дополнительно будут рассчитаны значения ошибки. Нажатие кнопки "Прогон сети" (Simulate Network) запишет результаты прогона в переменную, имя которой указано в поле "Выходы" (Outputs). Теперь можно вернуться в основное окно NNTool и, выделив мышью выходную переменную network1, нажать кнопку "Просмотр" (View). Содержимое окна просмотра совпадает со значением вектора целей - сеть работает правильно.

Следует заметить, что сеть создаётся инициализированной, то есть значения весов и смещений задаются определённым образом. Перед каждым следующим опытом обучения обычно начальные условия обновляются, для чего на вкладке "Инициализация" (Initialize) предусмотрена функция инициализации. Так, если требуется провести несколько независимых опытов обучения, инициализация весов и смещений перед каждым из них осуществляется нажатием кнопки "Инициализировать веса" (Initialize Weights).

Вернёмся к вкладке "Необязательная информация" (Optional Info) (рис. 8). Чтобы понять, какой цели служат представленные здесь параметры, необходимо обсудить два понятия: переобучение и обобщение.

При выборе нейронной сети для решения конкретной задачи трудно предсказать её порядок. Если выбрать неоправданно большой порядок, сеть может оказаться слишком гибкой и может представить простую зависимость сложным образом. Это явление называется переобучением. В

случае сети с недостаточным количеством нейронов, напротив, необходимый уровень ошибки никогда не будет достигнут. Здесь налицо чрезмерное обобщение.

Для предупреждения переобучения применяется следующая техника. Данные делятся на два множества: обучающее (Training Data) и контрольное (Validation Data). Контрольное множество в обучении не используется. В начале работы ошибки сети на обучающем и контрольном множествах будут одинаковыми. По мере того, как сеть обучается, ошибка обучения убывает, и, пока обучение уменьшает действительную функцию ошибки, ошибка на контрольном множестве также будет убывать. Если же контрольная ошибка перестала убывать или даже стала расти, это указывает на то, что обучение следует закончить. Остановка на этом этапе называется ранней остановкой (Early stopping).

Таким образом, необходимо провести серию экспериментов с различными сетями, прежде чем будет получена подходящая. При этом чтобы не быть введенным в заблуждение локальными минимумами функции ошибки, следует несколько раз обучать каждую сеть.

Если в результате последовательных шагов обучения и контроля ошибка остаётся недопустимо большой, целесообразно изменить модель нейронной сети (например, усложнить сеть, увеличив число нейронов, или использовать сеть другого вида). В такой ситуации рекомендуется применять ещё одно множество - тестовое множество наблюдений (Test Data), которое представляет собой независимую выборку из входных данных. Итоговая модель тестируется на этом множестве, что даёт дополнительную возможность убедиться в достоверности полученных результатов. Очевидно, чтобы сыграть свою роль, тестовое множество должно быть использовано только один раз. Если его использовать для корректировки сети, оно фактически превратится в контрольное множество.

Установка верхнего флажка (рис. 8) позволит задать контрольное множество и соответствующий вектор целей (возможно, тот же, что при обучении). Установка нижнего позволяет задать тестовое множество и вектор целей для него.

Обучение сети можно проводить в разных режимах. В связи с этим, в NNTool предусмотрено две вкладки, представляющие обучающие функции: рассмотренная ранее вкладка Train и "Адаптация" (Adapt). Adapt вмещает вкладку информация адаптации (Adap-tion Info), на которой содержатся поля, схожие по своему назначению с полями вкладки Training Info и выполняющие те же функции и вкладку параметры адаптации (Adaption Parameters). Последняя содержит единственное поле "Проходы" (passes). Значение, указанное в этом поле, определяет, сколько раз все входные векторы будут представлены сети в процессе обучения.

Параметры вкладок "Train" и "Adapt" в MATLAB используются функциями train и adapt, соответственно.

Разделение линейно-неотделимых множеств

Рассмотренная задача синтеза логического элемента "И" может трактоваться как задача распознавания линейно отделимых множеств. На практике же чаще встречаются задачи разделения линейно неотделимых множеств, когда применение персептронов с функцией активации с жёстким ограничением не даст решения. В таких случаях следует использовать иные функции активации.

Показательным примером линейно неотделимой задачи является создание нейронной сети, выполняющей логическую функцию "исключающее ИЛИ".

Пример 2

Пусть требуется создать нейронную сеть, выполняющую логическую функцию "исключающее ИЛИ".

Рассмотрим таблицу истинности этой функции (табл. 2).

Таблица 2. Таблица истинности логической функции "исключающее ИЛИ"

P1	P2	A
0	0	0
0	1	1
1	0	1
1	1	0

Что же подразумевается под "линейной неотделимостью" множеств? Чтобы ответить на этот вопрос, изобразим множество выходных значений в пространстве входов (рис. 10), следуя следующему правилу: сочетания входов P1 и P2, при которых выход A обращается в нуль, обозначаются кружком, а те, при которых A обращается в единицу - крестиком.

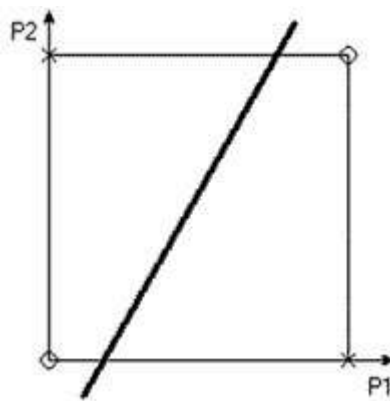


Рисунок 10. Состояния логического элемента "исключающее ИЛИ"

Наша цель - провести границу, отделяющую множество нулей от множества крестиков. Из построенной картины на рис. 10 видно, что невозможно провести прямую линию, которая бы отделила нули от единиц. Именно в этом смысле множество нулей линейно неотделимо от множества единиц, и персептроны, рассмотренные ранее, в принципе, не могут решить рассматриваемую задачу.

Если же использовать персептроны со специальными нелинейными функциями активации, например, сигмоидными, то решение задачи возможно.

Выберем персептрон с двумя нейронами скрытого слоя, у которых функции активации сигмоидные, и одним выходным нейроном с линейной функцией активации (рис. 11). В качестве функции ошибки укажем MSE (Mean Square Error - средний квадрат ошибки). Напомним, что функция ошибки устанавливается в окне "Создание сети" после выбора типа сети.

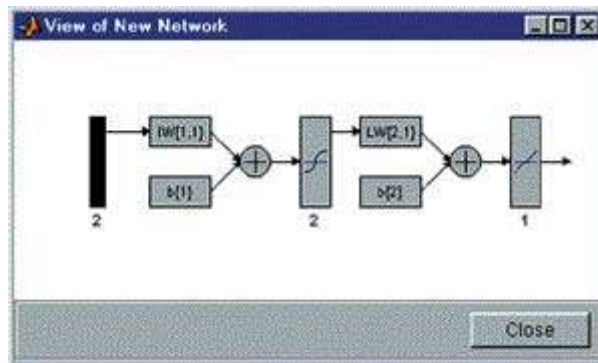


Рисунок 11. Сеть для решения задачи "исключающего ИЛИ"

Инициализируем сеть, нажав кнопку Initialize Weights на вкладке Initialize, по-сле чего обучим, указав в качестве входных значений сформированную ранее переменную data1, в качестве целей - новый вектор, соответствующий желаемым выходам. В процессе обучения сеть не может обеспечить точного решения, то есть свести ошибку к нулю. Однако получается приближение, которое можно наблюдать по кривой обучения на рис. 12.

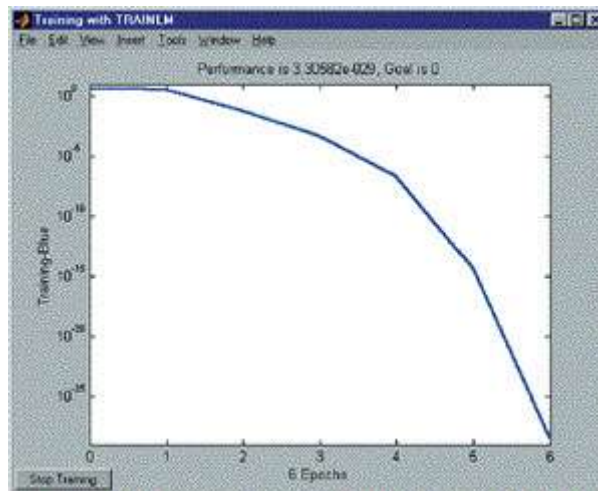


Рисунок 12. Кривая обучения в задаче "исключающего ИЛИ"

Следует отметить, что данная кривая может менять свою форму от эксперимента к эксперименту, но, в случае успешного обучения, характер функции будет монотонно убывающим.

В результате обучения, ошибка была минимизирована до весьма малого значения, которое практически можно считать равным нулю.

Задача синтеза элемента "исключающего ИЛИ" является также примером задачи классификации. Она отражает общий подход к решению подобного рода задач.

Задача аппроксимации

Одним из самых замечательных свойств нейронных сетей является способность аппроксимировать и, более того, быть универсальными аппроксиматорами. Сказанное означает, что с помощью нейронных цепей можно аппроксимировать сколь угодно точно непрерывные функции многих переменных. Рассмотрим пример.

Пример 3.

Необходимо выполнить аппроксимацию функции следующего вида

$$\sin\left(\frac{5\pi x}{N} + \sin\left(\frac{7\pi x}{N}\right)\right),$$

где $x \in 1:N$, а N - число точек функции.

Заготовим целевые данные, введя в поле "Значение" (Value) окна создания новых данных выражение:

$$\sin(5*\pi*[1:100]/100 + \sin(7*\pi*[1:100]/100)).$$

Эта кривая представляет собой отрезок периодического колебания с частотой $5\pi/N$, модулированного по фазе гармоническим колебанием с частотой $7\pi/N$ (рис. 15).

Теперь заготовим набор обучающих данных (1, 2, 3, ..., 100), задав их следующим выражением:

1:100.

Выберем персептрон (Feed-Forward Back Propagation) с тринадцатью сигмоидными (TANSIG) нейронами скрытого слоя и одним линейным (PURELIN) нейроном выходного слоя. Обучение будем производить, используя алгоритм Левенберга-Маркардта (Levenberg-Marquardt), который реализует функция TRAINLM. Функция ошибки - MSE. Полученная сеть имеет вид, изображённый на рис. 13.

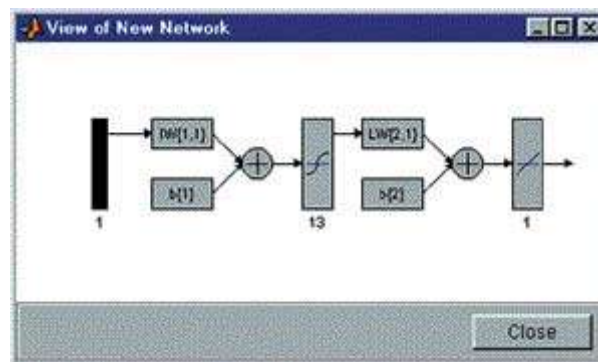


Рисунок 13. Архитектура сети для решения задачи аппроксимации

Теперь можно приступить к обучению. Для этого необходимо указать, какие наборы данных должны быть использованы в качестве обучающих и целевых, а затем провести обучение (рис. 14).

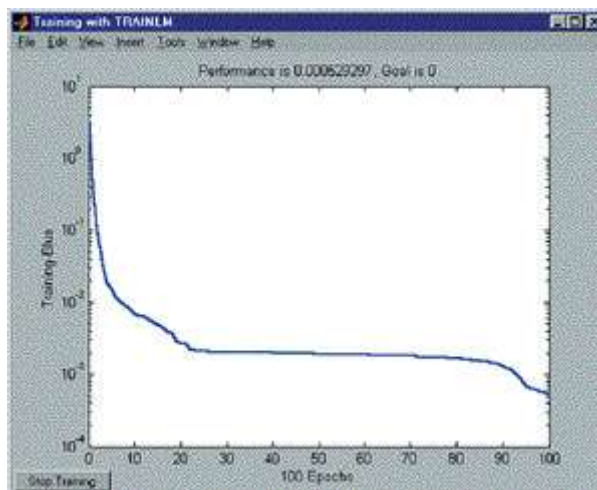


Рисунок 14. Кривая обучения в задаче аппроксимации

Рис. 15 иллюстрирует разницу между целевыми данными и полученной аппроксимирующей кривой.

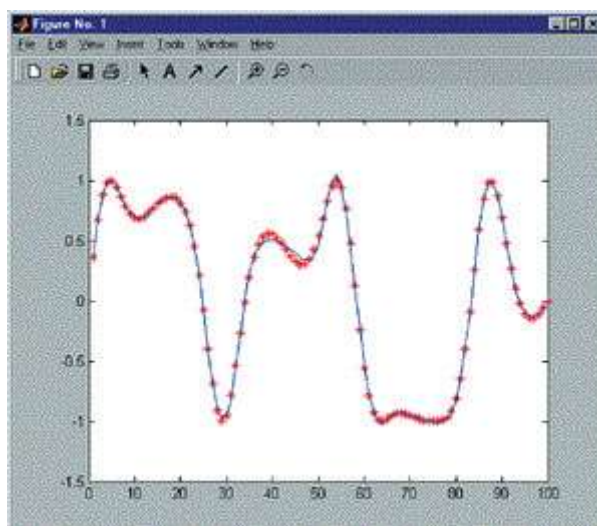


Рисунок 15. Красная кривая - целевые данные, синяя кривая - аппроксимирующая функция

Из рис. 14 и 15 видно, насколько уменьшилась ошибка аппроксимации за 100 эпох обучения. Форма кривой обучения на последних эпохах говорит также о том, что точность приближения может быть повышена.

Распознавание образов

Одной из популярных областей применения нейронных сетей является распознавание образов. Аппроксимационные возможности NN играют здесь первостепенную роль.

Пример 4.

Необходимо построить и обучить нейронную сеть для решения задачи распознавания цифр.

Цель этого примера - показать общий подход к решению задач распознавания образов. Поэтому, чтобы не загромождать изложение техническими деталями, обозначим лишь ключевые моменты, полагая, что на данном этапе читатель готов самостоятельно произвести соответствующие действия с NNTool.

Система, которую предстоит синтезировать с помощью нейронной сети, будет обучена воспринимать символы, близкие к шаблонным (рис. 16).



Рисунок 16. Монохромное изображение исходных данных

Наборы исходных данных (не путать с целевыми данными) можно как загружать из файлов изображений, так и создавать непосредственно в MATLAB. Рис. 17 иллюстрирует содержимое массива исходных данных.

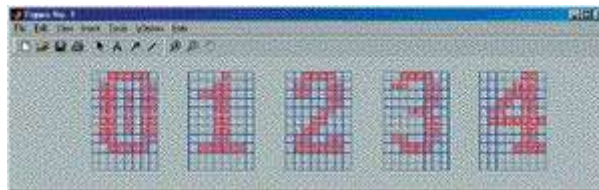


Рисунок 17. Графическое представление массива исходных данных

Получим обучающие данные, наложив шум на набор исходных данных (рис. 18).

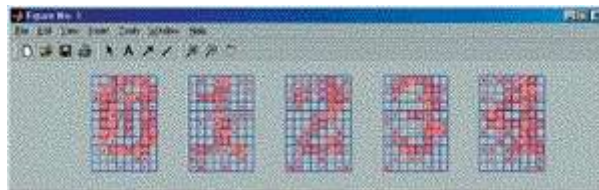


Рисунок 18. Обучающие данные

В задачах классификации, к которым относится данный пример, количество выходов сети соответствует числу разделяемых сетью классов. Этот факт должен быть учтён при выборе архитектуры сети и на этапе формирования целевых данных.

Сеть классификации даёт наибольшее значение на выходе, который соответствует подходящему классу. При хорошо сконструированной и обученной NN значения других выходов будут заметно меньше.

Для решения этой задачи выбрана сеть Feed-forward backprop с пятью сигмоидными нейронами первого слоя и пятью линейными нейронами второго слоя. Алгоритм обучения - Левенберга-Маркардта. С такой конфигурацией сеть после восьми эпох обучения дала ошибку порядка 10^{-30} .

Чтобы удостовериться в правдивости результата, мы прогнали сеть на заготовленном контрольном множестве. Сеть безукоризненно разделила и новую выборку зашумлённых символов.

Завершая пример, отметим, что уже сейчас успешно разрабатываются разнообразные системы распознавания образов (визуальных, звуковых), базирующиеся на нейронных сетях. Примером могут послужить программы распознавания сканированного текста. Они до-статочно широко

распространены среди обладателей оптических сканеров и в реальном масштабе времени справляются с поставленными задачами.

Импорт-экспорт данных

На практике часто приходится переносить данные с одного компьютера на другой или пользоваться внешними, по отношению к NNTool, средствами обработки данных. В связи с этим возникает необходимость сохранения результатов работы и загрузки данных. Не менее важен обмен данными между NNTool и MATLAB, поскольку пространства их переменных не пересекаются.

Эти задачи решают средства импорта-экспорта и загрузки-сохранения данных. Доступ к ним осуществляется через главное окно NNTool (рис. 1) посредством кнопок Import и Export.

Импорт

Источником служит переменная в рабочем пространстве MATLAB, а пунктом назначения - переменная в рабочем пространстве NNTool. Нажав кнопку Import, попадаем в окно "Импорта-загрузки данных" (Import or Load to Network/Data Manager). По умолчанию здесь установлена опция загрузки из рабочего пространства MATLAB, поэтому в центре окна появляется список принадлежащих ему переменных. Выбрав мышью нужную переменную в поле "Выбор переменной" (Select a Variable), - ей можно задать произвольное имя в поле "Имя" (Name), под которым она будет скопирована в NNTool. Когда переменная выделена, NNTool анализирует её тип и делает доступными для выбора те "Категории данных" (Import As), которые поддерживаются. Указав одну из них, следует нажать кнопку "Импортировать" (Import), чтобы завершить копирование. После того, как все действия успешно проведены, имя импортируемой переменной появится в одном из списков главного окна NNTool.

Загрузка из файла

Здесь источник - файл. При этом важно, чтобы его формат поддерживался NNTool. В подходящем формате хранятся так называемые MAT-файлы. Они содержат бинарные данные и позволяют MATLAB сохранять переменные любых поддерживаемых типов и размерностей. Такие файлы могут создаваться, например, в процессе работы с NNTool. Чтобы загрузить переменные из MAT-файла в NNTool, необходимо открыть окно импорта, нажав кнопку "Import" главного окна NNTool. Затем следует отметить опцию "Загрузить из файла" (Load from disk file) и, нажав "Обзор" (Browse), открыть файл с данными, хранящимися в формате MAT-файлов. Чаще всего, это файлы с расширением MAT. В результате, список в окне импорта заполнится именами переменных, сохранённых в указанном MAT-файле. Дальнейшая последовательность действий полностью совпадает с описанной в пункте Импорт (рис. 19).

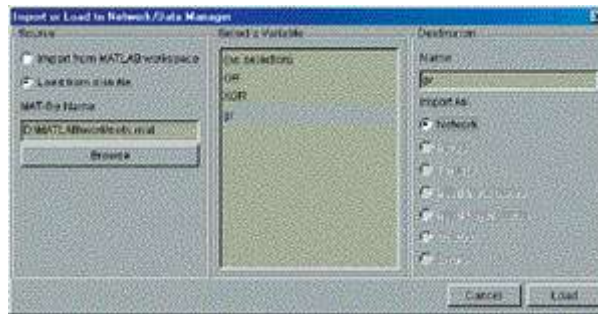


Рисунок 19. Окно импорта и загрузки данных из MAT-файла

Экспорт

Эта функция копирует заданные переменные из NNTool в рабочее пространство MATLAB. Она доступна по нажатию кнопки Export главного окна NNTool. В окне экспорта единым списком перечислены переменные всех категорий, представленных в NNTool. Здесь необходимо выделить те из них, которые подлежат экспорту, и нажать Export. Теперь выделенные переменные скопированы в рабочее пространство MATLAB.

Сохранение в файле

В целом, процедура схожа с экспортом, за одним исключением: отметив переменные, следует нажать кнопку "Сохранить" (Save). Тогда появится окно, в котором можно задать имя файла. Его можно указать без расширения - по умолчанию NNTool прикрепит расширение MAT. Это связано с тем, что NNTool сохраняет данные только в формате MAT-файлов (рис. 20).

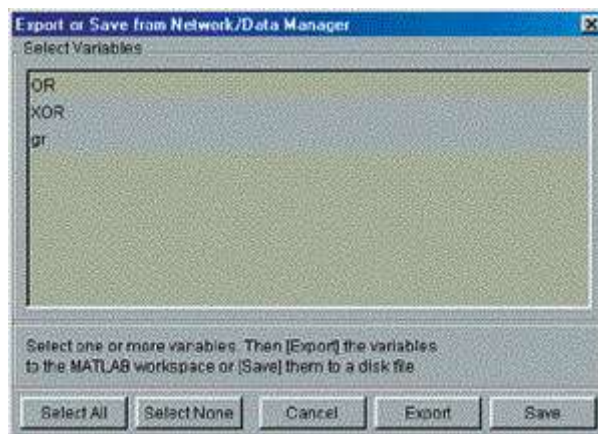


Рисунок 20. Окно экспорта и сохранения данных в MAT-файле

Мы рассмотрели простейшие задачи синтеза цепей и обработки сигналов, для решения которых применялись наиболее популярные нейронные сети прямого распространения. На самом деле, NNTool позволяет решать значительно более широкий круг задач, предоставляя возможность использовать сети разнообразных архитектур, с памятью и без, с обратными связями и без таковых. При этом следует иметь в виду, что успех во многом зависит от понимания поведения конструируемых сетей и их аппроксимационных возможностей.

Литература

1. Demuth H., Beale M. Neural Network Toolbox. For Use with MATLAB. The MathWorks Inc. 1992-2000.
2. Нейронные сети. STATISTICA Neural Networks. Москва. Горячая линия – Телеком. 2000.
3. Principe J.C., Euliano N.R., Lefebvre W.C. Neural and Adaptive Systems. Fundamentals Through Simulations. New York. John Wiley & Sons Inc. 2000.
4. Luo F-L., Unbehauen R. Applied Neural Networks for Signal Processing. Cambridge University Press. 1998.